XCELL

THE NEWSLETTER FOR XILINX PROGRAMMABLE LOGIC USERS

Issue 9 Second Quarter 1993

XCELL must appear more regularly, and must reach every Xilinx user. This is a clear message from the Xilinx user community. In response, we now ship "The Best of XCELL, #1 through #9" with all development systems and updates.

In the future, we will make sure that XCELL appears every quarter, bringing you updates, ideas and tutorials about the use of Xilinx devices and development systems.

New products appear at an increasing pace, and XCELL is here to explain them to you.

Peter Alfke, Editor

Table of Contents

1992 Was a Good Year

Xilinx sales increased from $130M in calendar 1991 to $163 M in 1992, maintaining a solid position as the largest manufacturer of all types of CMOS Programmable Logic, well ahead of AMD and Altera. We introduced new products at an ever increasing pace. In 1992, we doubled our product offerings without counting speed and package options:

We started 1992 with 16 different devices, and we ended 1992 with 32 different devices.

We expect to double the breadth of our product offering again in 1993; in the first four months of 1993 we have already introduced seven new devices.

Programmable Logic is no longer a niche product line used for prototyping. Prices have come down, speed and density have increased, and our customers appreciate the advantages of a shorter development cycle and faster time-to-market more than ever.

Number of Available Device Types

Family	JAN'92	JAN'93	APR'93
XC2000	2	2	4
XC3000	5	5	7
XC4000	3	8	10
XC7200	0	2	3
XC17000	2	3	3
HardWire	4	6	6
Total	**16**	**32**	**39**

Programmable logic is, therefore, now being used in volume production. That, in turn, puts pressure on Xilinx to reduce cost and prices, and to increase performance and density even more.

One result of this pressure is product diversification. One product family, or even one technology, cannot possibly cover all bases. No single product family can simultaneously be best in cost, speed, and density for all applications.

The future will see a greater diversity of programmable logic device architectures and technologies. Different families will address different issues. Some will emphasize low cost and high density, sacrificing speed. Some families will be ultra-fast, but more expensive. The world is already familiar with the difference between the register-rich FPGA architecture, and the more structured and predictable EPLD architecture. Expect additional architectures and technologies to address different application areas, and expect Xilinx to remain the leader in this innovation.

Programmable logic has become a significant, and fast growing part of the electronics industry. Xilinx is totally dedicated to Programmable Logic, in any practical architecture and technology. XCELL will keep you informed about new developments.

PA

Component Availability (May1993)

✓ Product currently shipping or planned

PINS	44	48	64	68		84		100				120	132		144	156	160	164	175		176	191	196	208		223	240	
TYPE	PLAST. PLCC	PLAST. DIP	PLAST. VQFP	PLAST. PLCC	CERAM. PGA	PLAST. PLCC	CERAM. PGA	PLAST. PQFP	PLAST. TQFP	PLAST. VQFP	TOP-BRAZED CQFP	CERAM. PGA	PLAST. PGA	CERAM. PGA	PLAST. TQFP	CERAM PGA	PLAST. PQFP	TOP-BRAZED CQFP	PLAST. PGA	CERAM. PGA	PLAST. TQFP	CERAM. PGA	TOP BRAZED CQFP	PLAST. PQFP	METAL MQFP	CERAM. PGA	PLAST. PQFP	METAL MQFP
CODE	PC44	PD48	VQ64	PC68	PG68	PC84	PG84	PQ100	TQ100	VQ100	CB100	PG120	PP132	PG132	TQ144	PG156	PQ160	CB164	PP175	PG175	TQ176	PG191	CB196	PQ208	MQ208	PG223	PQ240	MQ240
XC2064	✓	✓		✓	✓																							
XC2018	✓		✓	✓		✓	✓		✓																			
XC2064L			✓	✓																								
XC2018L			✓			✓				✓																		
XC3020				✓		✓	✓	✓			✓																	
XC3030	✓			✓		✓	✓	✓	✓																			
XC3042						✓	✓	✓	✓		✓		✓	✓														
XC3064						✓							✓	✓			✓											
XC3090						✓											✓	✓	✓	✓				✓				
XC3020A				✓		✓	✓	✓																				
XC3030A	✓		✓	✓		✓	✓	✓		✓																		
XC3042A						✓	✓	✓		✓			✓	✓	✓													
XC3064A						✓							✓	✓	✓		✓											
XC3090A						✓											✓		✓	✓	✓			✓				
XC3020L			✓			✓																						
XC3030L			✓			✓				✓																		
XC3042L						✓				✓					✓													
XC3064L						✓									✓													
XC3090L						✓															✓							
XC3120				✓		✓	✓	✓			✓																	
XC3130	✓			✓		✓	✓	✓	✓																			
XC3142						✓	✓	✓	✓		✓		✓	✓	✓													
XC3164						✓							✓	✓			✓											
XC3190						✓											✓	✓	✓	✓				✓				
XC3195						✓											✓		✓	✓				✓		✓		
XC4005						✓										✓	✓	✓						✓				
XC4006																✓	✓							✓				
XC4008																						✓		✓	✓			
XC4010																						✓	✓	✓	✓			
XC4013																									✓	✓	✓	✓
XC4002A						✓		✓		✓		✓																
XC4003A						✓		✓		✓	✓	✓																
XC4004A						✓						✓			✓		✓											
XC4005A						✓									✓	✓	✓							✓				
XC4003H																						✓		✓				
XC4005H																										✓	✓	✓

Software Availability (May1993)

Code	Software Product	
DS343	Mentor Schematic and Simulation Interface V8	V4.10
DS344	Mentor Schematic and Simulation Interface V7	V1.01
DS35	OrCAD Schematic and Simulation Interface	V4.22
DS390	Viewlogic Viewdraw-LCA Editor, Libraries and Interface	V4.22
DS391	Interface and Libraries for Viewlogic Workview for PC	V4.15
	for Workstations	V5.01

Code	Software Product	
DS371	Xilinx ABEL	V1.03
DS401	Synopsys Interface and Libraries	V2.0
DS380	X-BLOX™ Architectural Synthesis	V1.04
DS355	OrCAD Schematic and Simulation Interface	V3.11
DS502	FPGA XACT Design Implementation	V1.41
DS550	EPLD Design Implementation	V4.0
DS290	Viewlogic Viewsim Simulator and Interface	V4.15

A Flood of New Programmable Logic Devices

Hardly a week passes, it seems, without the introduction of a new programmable logic device, complete with its block diagram and new acronyms to describe its features. How much longer until this bewildering flood of new PLD products abates and the industry settles down with a few comfortable standards? Before getting into the outlook for programmable logic, let's see what we can learn from the evolution of memories and microprocessors.

Memories and Microprocessors

Memories are the simplest case because most memory applications are nearly identical. Standard requirements can be met with standard memory architectures that are offered by all major suppliers. The designer's menu of alternatives is pretty simple — EPROMs or FLASH for permanent storage, SRAMs for high speed, DRAMs for low cost, and a few application-specific options like video RAMS. Within each of these categories, new product generations apply advances in process technology to existing architectures. With improved speed, density, and cost, each new generation quickly supplants its predecessors, keeping things simple.

The picture for microprocessors is much the same. In terms of the memory and I/O environment surrounding the microprocessor, most applications are very similar. Unlike memories, however, microprocessors come in different architectural "flavors" — Intel, Motorola, RISC, etc. Occasionally a design engineer will have the opportunity to evaluate these different alternatives for optimum performance in his application. Far more often, however, the choice is dictated by software compatibility requirements. New microprocessor generations preserve software compatibility while taking advantage of more advanced processes to get more integration and higher speed. Each new generation quickly displaces older products in new designs, holding down the growth in the number of options.

Microcontrollers are another story. Embedded control applications vary widely, and no single microcontroller architecture provides the best solution for every one. The range of applications is far too broad for a single device with a superset of all possible features to be cost effective. As a result microcontroller product lines continue to grow. Today Motorola alone offers four basic microcontroller architectures and a total of more than 60 different microcontroller types, each optimized to fit the speed, memory, I/O, and cost requirements of a particular class of applications.

Microcontrollers are often used as an alternative way of implementing low speed logic functions, so it's not surprising that the evolution of programmable logic is more like microcontrollers than either memory or microprocessors.

Logic

Logic performs the non-memory functions that require more performance than a microprocessor or microcontroller can provide. Each logic application tends to be unique. Logic is the portion of a design that differentiates it from similar systems supplied by competitors, each of which is probably using the same memory and microprocessor. The thickness of a TTL catalog is ample testament to the varied nature of logic requirements. VLSI can integrate logic functions and reduce the number of logic chips on a board, but VLSI cannot reduce the number of logic chip *types*. Of course if we only counted base array types, gate arrays reduce the number of logic chip types dramatically. From a production, testing, purchasing, and manufacturing perspective, however, gate arrays and other custom solutions result in a significant proliferation of logic chip types. Each design is a unique IC.

All we need to dramatically reduce the number of IC types that a system designer has to worry about is the user-programmable equivalent of a gate array. Such a chip would meet the speed, density, and cost requirements of all digital logic applications with a small number of device types. What's the problem? In a word, *programmability*. We don't know now, and we may never know, how to deliver user programmability with the efficiency of a mask-programmed gate array. No user-programmable circuit can match the speed or the density of the vias used to program a gate array. The only way programmable devices can overcome this handicap is through innovation in *architecture*.

Programmable Logic

In order to dramatically increase the number of applications where programmable devices can be used, advances in speed and cost must be revolutionary. Evolutionary improvements from new semiconductor processes will help, of course, especially in narrowing the cost gap. Still the only hope for doubling speed and halving cost

(continued from page 3)

relative to the best existing devices, assuming the same manufacturing process, is through improved programmable logic architectures. And that will mean more programmable device types.

Programmable logic comes in three flavors: simple PLDs (SPLDs), complex PLDs (CPLDs), and field programmable gate arrays (FPGAs). Each of these three forms exists in a number of different product types. We may already be near the optimal architecture for SPLDs. This category appears to be showing some signs of architectural consolidation. Flexible new architectures like the Lattice GAL can replace most commonly used SPLDs with a single device type.

The problems for CPLDs and FPGAs are considerably more difficult. Yet despite the differences between these two types, the underlying architectural issues are remarkably similar.

Complex PLDs

The AND-OR plane SPLD architecture has become an industry standard logic solution, widely used and familiar to many design engineers. Its shortcomings are density and cost. Altera's first CPLD addressed these with an architecture that includes on-chip interconnections which ties together a number of SPLD-like structures. CPLDs are now available in a growing list of varieties from a growing list of suppliers.

The fundamental architectural issue for CPLDs is the nature of the interconnection matrix, which grows in complexity as the square of the number of elements to be interconnected. Although the density offered by new manufacturing process would easily permit the addition of more logic blocks, four times as many logic blocks would make the interconnection matrix sixteen times larger and significantly slower. The basic CPLD architecture cannot be extended without some modifications. One modification that is widely used limits the interconnect alternatives for each logic block. This permits the addition of more logic blocks, but both utilization and performance become more application dependent. Another alternative is to increase the complexity of each logic block. This can also reduce utilization if logic functions cannot be partitioned efficiently across these larger granules. A third alternative is to replace some of the general purpose logic blocks with special purpose functions such as RAM or other application specific logic. If defined properly, such devices will offer superior levels of speed and cost for the targeted applications. All of these architectural options have merit, but their use will inevitably result in more CPLD device types.

Field Programmable Gate Arrays

The first FPGAs were developed by Xilinx, and are now available from a growing list of suppliers. There are only three architectural elements — logic blocks, interconnection resources, and I/O blocks. The primary advantage of FPGAs over CPLDs is the ability to scale this basic architecture to larger array sizes with advances in process technology. At a more detailed level, however, the architectural challenges are similar to those of CPLDs. For FPGAs the fundamental architectural issue is the balance between the complexity or granularity of the logic block and the routing resources required to ensure good utilization. Table 1 compares logic granularity, measured by usable gates per logic block, for some currently available FPGA architectures.

As shown in Table 1, the architectures available from these FPGA suppliers exhibit a 40:1 range in logic block granularity. Although this range may shrink as FPGA architectures become more mature, there is no evidence that there is any *best* solution that will eventually displace all others.

Attribute-Specific Programmable Logic

Most, if not all, of the FPGA architectures summarized in Table 1 were designed to be general purpose, balancing the trade-offs among speed, cost, density, and ease of use. As shown in Figure 3, Xilinx's first three FPGA generations were all aimed at a compromise among these conflicting requirements. The resulting architectures have met the needs of many designers, and have created a wide applications base for FPGAs.

Table 1: Usable Gates per Logic Block

Device	Gates/Block	Program Element
Actel ACT 2	5	Anti-fuse
Altera FLEX	10	SRAM
Crosspoint CP202200	0.5	Anti-fuse
Plessey ERA (Pilkington)	0.5	SRAM
Quicklogic QL8X12	10	Anti-fuse
Xilinx XC3000	12	SRAM
Xilinx XC4000	20	SRAM

The mainstream FPGA market will continue to be dominated by general purpose devices for some time. It will take more, however, than another evolutionary extension of a general purpose FPGA architecture to double speed or halve cost. The overhead associated with programmability is already high. Compromises and design tradeoffs further encumber FPGA designs and reduce their performance.

In order to create the significant improvements in speed and cost that will permit the use of FPGAs in many new applications, it will be necessary to develop new architectures that depart from the general purpose model of Figure 3. These new FPGAs will not necessarily be *application* specific. Most new FPGA generations will still be used across a wide range of end use systems. Rather FPGAs will become *attribute* specific. The difference is shown in Figure 4. In some cases speed will be emphasized over all other considerations. In other architectures, cost considerations will be given the highest weight in making design decisions.

Process Technology

There is not yet a "best" process technology for programmable logic. Nor is there one on the horizon. The choice of the process technology used to implement the programming element, however, does influence logic block granularity. Smaller programming elements such as anti-fuses can support smaller logic blocks, while most SRAM-based architectures use larger blocks. Architectures with smaller blocks tend to be more flexible, while those with larger blocks tend to have advantages in speed and cost. Due to the link between process and architectures, more process technologies will mean more programmable logic device types.

The advantages and disadvantages of the various programmable logic process technologies seem to be of little interest outside the narrow confines of the programmable logic industry. The general trend to smaller geometries, however, has important cost implications for all high-density programmable logic. Since these chips tend to be big, they benefit far more than conventional gate arrays in the cost improvement provided by each new process generation.

The Role of Software

How is an engineer to evaluate all the available programmable architectures and find the best solution for a system design problem? One imminent solution is a set of standard benchmarks for speed and density under development by the programmable logic industry. But this still requires that the designer understand at least the rudiments of each architecture, and also evaluate how closely published benchmarks reflect the needs of a particular real-life application.

The long term answer is more powerful development system software for programmable logic. Software now in development will not only suggest the best architecture for a particular application, based on parameters such as speed, package type, and cost, but it will also partition large designs across a range of potentially different programmable logic devices. The proliferation of programmable logic device types may not slow, but software will make the architectural details less obtrusive.

Summary

User programmable logic is an immature, emerging technology. While the architectural evolution of microprocessors and microcontrollers was guided by the work done earlier for mainframes and minicomputers, there is no such architectural precedent for programmable logic. Another difference between programmable logic and microprocessors is the lack of user-visible software. The requirement for software compatibility, which has undoubtedly inhibited creativity in the development of new computer and microprocessor architectures, does not limit innovation in this new field.

A high rate of new discoveries implies substantial R&D investments in both programmable IC architectures and the software to support them. This is likely to lead to further consolidation among the companies participating in this business. The remaining companies will offer very broad product lines with common software support, and will continue to introduce improved programmable logic devices at a high rate, at least for the next few years. More new FPGA architectures will be introduced in the next two years than we have seen since the announcement of the first FPGA seven years ago.

All this is good news for the system designer. Fewer and fewer applications will require speed, density, or cost that dictate a custom IC solution. More designers will enjoy the flexibility of user programmability, and some will add in-system reprogrammability to their repertoire of design techniques. Continuing improvements in programmable logic software will hide most of the details of the new architectures, leaving the designer with just the benefits.

Wes Patterson

Reprinted with kind permission from Computer Design, February 1993

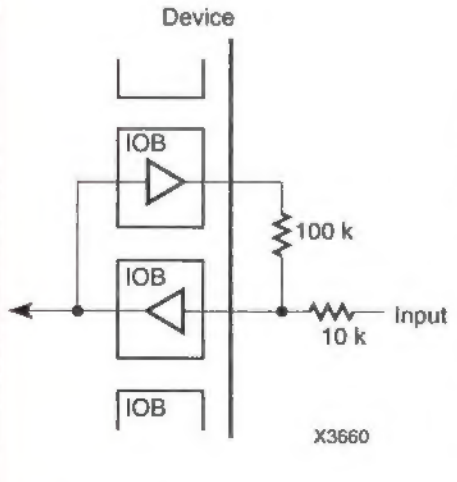
User-Defined Schmidt Trigger

All inputs to Xilinx LCAs have a hysteresis of 100 to 200 mV, which helps avoid noise propagation from slowly rising or falling input signals. Some users want more hysteresis. They want to design their own Schmidt trigger circuit. This can easily be done with only two resistors, but it uses one additional output pin.

A 10 kΩ serial input resistor combined with a 100 kΩ feedback resistor gives 500 mV of hysteresis (Hysteresis is the difference between the effective input threshold voltages on the Low-to-High transition and the High-to-Low transition. That difference is equal to Vcc times the resistor ratio).

A 1 kΩ / 10 kΩ combination gives the same hysteresis, but reduces the delay caused by the pin-to-ground capacitance from 100 ns to 10 ns. On the other hand, it requires more input current, i.e. a lower drive impedance.

For TTL-threshold inputs, the hysteresis should be kept below 1 V, for CMOS inputs, it can be up to 2 V.



Schmidt-Trigger Input

XC4000 Program Pin, Function and Timing

Some users have requested function and timing details about the $\overline{\text{PROGRAM}}$ pin, when used to re-configure the XC4000 device. The XC4000 data book has this information hidden in two places, page 47(upper right corner) and page 27 (upper left corner).

Any active Low pulse on $\overline{\text{PROGRAM}}$, provided it is wider than 300 ns, triggers a re-configuration at any time, irrespective of the present state of the part (during configuration or after).

DONE responds with a Low level within 300 ns, and the I/Os go 3-state at about the same time. All this is completely unconditional.

A Low level on $\overline{\text{PROGRAM}}$ keeps the device in the "Clear configuration memory" state. When $\overline{\text{PROGRAM}}$ has gone High again, and the clearing is finished, and the power-on time-out is no longer active, if we are re-configuring, the power-on time-out is obviously long past, then the memory is being cleared once more, and $\overline{\text{INIT}}$ is checked before configuration starts.

Hot and Cold

All Xilinx devices are specified over the 0°C to +70°C commercial temperature range. Many are also guaranteed over the -40°C to +85°C industrial and -55°C to +125°C military temperature ranges. But, that is still not enough for some users who operate their equipment at more exotic temperatures.

For many years, Xilinx FPGAs have been used in "downhole" applications, where they operate at the bottom of mile-deep oil-exploration bore holes. They are part of sophisticated instrumentation packages, consisting of microprocessors (often the venerable Z80), SRAMs and some rugged transducers and A/D converters. The operating temperature is >150°C or >300°F, and our FPGAs work just fine, although slower than their specification.

We recently heard about a cryogenic application, where an XC3090 works immersed in liquid nitrogen at -192°C or -314°F. It works fine and is very fast. The low-voltage detect threshold had changed from 3.1 V to 4.2 V, but our upcoming XC3090L has a lower detect threshold and will re-establish the safety margin even at this very cold temperature.

We do not guarantee reliability, parameters, or even functionality at these temperatures, but the designers of such exotic equipment don't ask for guarantees; they are happy to have access to working devices, and we are happy to serve science and industry in the heat and the cold. PA

Programmable Logic Device Benchmarks

Over the past years, many FPGA manufacturers have made exorbitant claims about the speed and density of their own products, usually depicting the competition as being vastly inferior. Some of these claims were deliberate deceptions, some were based on ignorance, a few may even have been correct, but all caused confusion in the user community. Which family is really the fastest, and which one is the densest for a given application?

To generate some order out of this chaos, the leading FPGA and EPLD manufacturers got together in 1991 to create reliable benchmarks that would give the potential user a good feel for the achievable speed and density of various device families. Xilinx, Actel, and Altera were the "founding fathers" of what became PREP*, the Programmable Electronics Performance Corporation, headed by Stan Baker, an editor of EETimes, and organizer of the annual PLD conference. AMD, Cypress, Intel, Lattice, Quicklogic, TI, and Data I/O joined later, and participated in the definition of benchmarking methodology and the first suite of nine benchmarks.

These are relatively simple circuits of a few hundred gate complexity. Each can be implemented many times in an FPGA or EPLD. The circuits are defined with eight inputs and eight outputs, so that they can be concatenated until they fill the device.

These PREP benchmarks will be honest and reliable, because they are implemented by the engineers who know their devices best, and they are then checked by the people most critical of any deceptive claims.

All benchmarks are implemented by the respective manufacturer, and have been checked for accuracy by any one or several of the competing manufacturers, or by PREP.

All benchmarks must be implemented in the largest member of any device family, but the manufacturer is free to add benchmarks for smaller family members. This rule was based on the assumption that large devices might be slower than small ones, not true for some FPGAs.

Each device family must report all nine benchmarks, even the ones that are unfavorable. (The large state machine is tough for FPGAs, while the prescaled counter shows relatively poor EPLD performance.) Each company must report results for "automatic" implementation, but can also add measurements for interactive ("manual") design.

Gate count was considered a meaningless and confusing number, not included in the report. These benchmarks focus on device density and performance; any commercially available development software can, therefore, be used, but run-time is not reported, since it is not a device characteristic, and changes too often with new software releases.

Conclusions

These benchmarks are the first successful attempt at an objective, standardized method of measuring the capacity and performance of Programmable Logic Devices. The results are assumed to be honest and correct, but there are still some questions about their relevance.

While these benchmarks may represent a fair mix of typical logic, the way identical instances are concatenated is entirely artificial. In order to measure the capacity of dissimilar architectures and a wide range of device sizes in a uniform way, the PREP committee agreed on a step-and-repeat methodology, where one circuit instance feeds the next. This bears no resemblance whatsoever to the normal use of Programmable Logic Devices. It also does not stress their capability to handle "spaghetti-logic" with random routing all over the chip.

The PREP benchmarks also do not address the system-oriented features available in some of the more refined PLDs, like:

- Multiple low-skew clock lines
- Internal 3-state drivers
- Dedicated input/output flip-flops
- Output 3-state control
- Output slew-rate control,
- Bus drive capability
- Boundary scan,
- Low power consumption
- In-circuit programmability,
- etc.

Some of these aspects may be covered by future benchmarks.

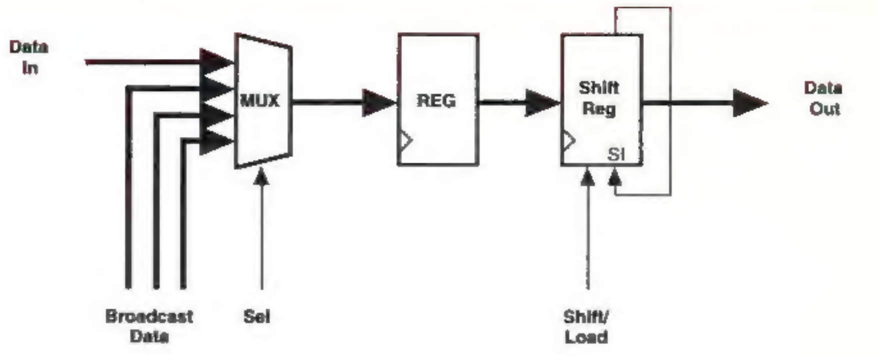
It is easy to criticize the present methodology. It is far more difficult to come up with a better alternative. Filling a chip to the rim with one function of growing width is not the solution. An 800-bit synchronous binary counter, or a 3600-bit parity checker in an XC4010 are possible, but they would not be a meaningful benchmark for any user. PREP will have to define additional meaningful benchmarks of higher complexity that give the competing architectures more freedom to demonstrate their respective strengths.

* PREP is a registered trademark of Programmable Electronics Performance Corporation

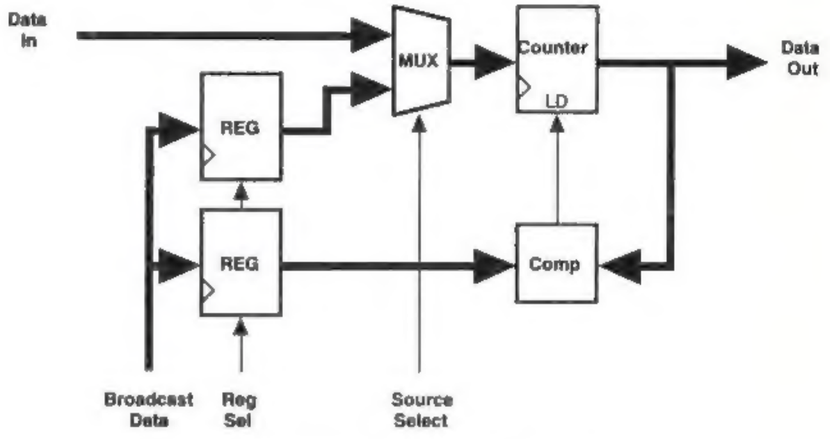
(continued from page 7)

The first suite of PREP benchmarks was created, refined, and debated (sometimes heatedly) by applications engineers from the various manufacturers, meeting and working together in a constructive way. It was Silicon Valley at its best, and it created a camaraderie and mutual respect that may help us in the future marketing battles.

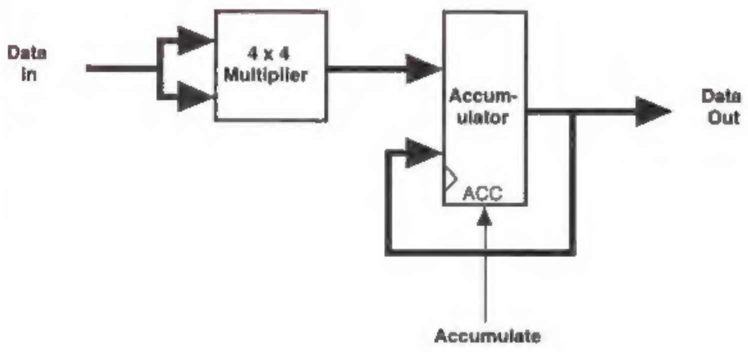
We tried very hard to create a tool set that is helpful to the user, not one that gives bragging rights to the manufacturers. Only the future will show whether we were successful. PA



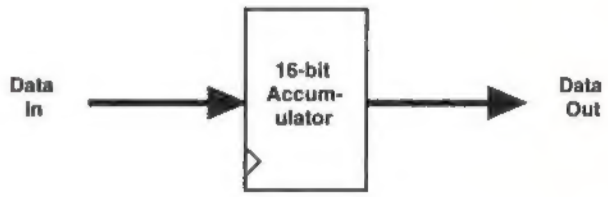
Benchmark No.1 Data Path



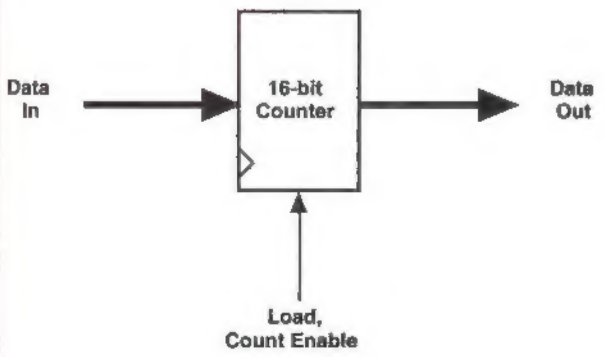
Benchmark No.2 Timer/Counter



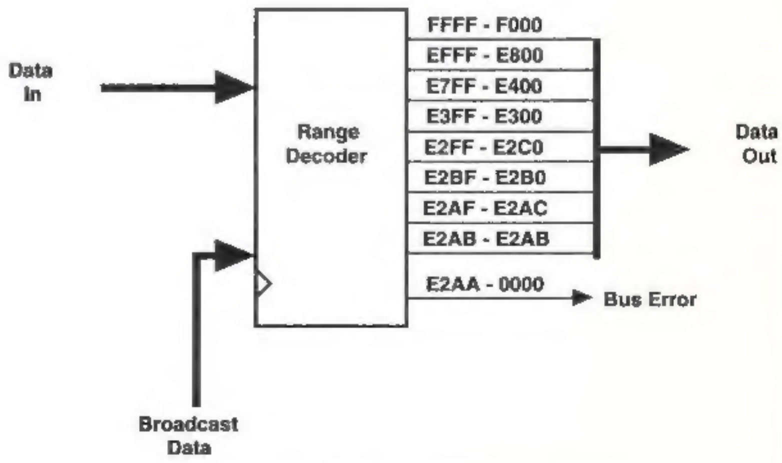
Benchmark No.5 Arithmetic Circuit



Benchmark No.6 16-Bit Accumulator



Benchmark No.7,8 16-Bit Counter



Benchmark No.9 Memory Mapper

Benchmark Results

Company	Device/Grade	Capacity: Reps	Fill	Rep%	Fill%	Performance (MHz): Worst	Best	Mean	Ext	Optimized: Cap	Perf	Tools Used: Vendor	Tool Name	Rev	APR	Crit	HM	Manp	Manr	NOTES
Benchmark #1																				
Xilinx	4010-5	30	0	90%	0%	42	71	58	[illegible]			Xilinx	PPR	1.3	X					
Xilinx	4010-5	30	0	90%	0%	70	[illegible]	83	39			Xilinx	PPR	1.3			X	X	X	1
Xilinx	3090-125	20	0	100%	0%	48	58	55	24			Xilinx	ADI	3.31				X		
Xilinx	3190-3	20	0	100%	0%	89	108	101	39			Xilinx	ADI	3.31				X		
Xilinx	3090-125	18	0	90%	0%	41	[illegible]	[illegible]	46			Xilinx	ADI	3.31	X	X				
Xilinx	3190-3	18	0	90%	0%	72	111	87	[illegible]			Xilinx	ADI	3.31	X					
Xilinx	7272-25	4	0	89%	0%	40	40	40	25			Xilinx	XEPLD	3.1	X					
Benchmark #2																				
Xilinx	3090-125	12	0	90%	0%	20	28	24	24			Xilinx	ADI	3.31	X					
Xilinx	3190-3	12	0	90%	0%	37	52	43	44			Xilinx	ADI	3.31	X					
Xilinx	4010-5	[illegible]	0	88%	0%	23	30	26	27			Xilinx	PPR	1.3	X	X				
Xilinx	4010-5	30	0	90%	0%	[illegible]	40	40	20			Xilinx	PPR	1.3			X	X	X	1
Xilinx	7272-25	2	0	75%	0%	20	20	20	20			Xilinx	XEPLD	3.1	X					
Benchmark #3																				
Xilinx	3090-125	16	0	80%	0%	20	35	27	24		X	Xilinx	ADI	3.31	X					
Xilinx	3090-125	20	0	100%	0%	15	[illegible]	24	23	X		Xilinx	ADI	3.31	X					
Xilinx	3190-3	16	0	80%	0%	36	61	49	39		X	Xilinx	ADI	3.31	X					
Xilinx	3190-3	20	0	100%	0%	25	52	41	37	X		Xilinx	ADI	3.31	X					
Xilinx	4010-5	36	0	99%	0%	28	47	39	26			Xilinx	PPR	1.3	X					
Xilinx	7272-25	8	0	100%	0%	40	40	40	25			Xilinx	XEPLD	3.1	X					
Benchmark #4																				
Xilinx	3090-125	9	0	96%	0%	17	18	14	14			Xilinx	ADI	3.31	X					
Xilinx	3190-3	9	0	96%	0%	11	30	[illegible]	23			Xilinx	ADI	3.31	X					
Xilinx	4010-5	15	0	98%	0%	27	30	27	20			Xilinx	PPR	1.3	X	X				
Xilinx	7272-25	1	0	89%	0%	40	40	40	13			Data I/O	XEPLD	3.1	X					
Benchmark #5																				
Xilinx	3090-125	12	0	86%	0%	12	17	14	14			Xilinx	ADI	3.31	X	X				
Xilinx	3190-3	12	0	86%	0%	20	29	24	23			Xilinx	ADI	3.31	X	X				
Xilinx	4010-5	19	0	100%	0%	18	20	19	15		X	Xilinx	PPR	1.3	X	X				
Xilinx	4010-5	[illegible]	0	100%	0%	19	20	19	16	X		Xilinx	PPR	1.3			X	X		
Xilinx	7272-25	1	2	54%	38%	n/a	n/a	n/a	10			Xilinx	XEPLD	3.1	X					
Xilinx	7272-25	1	2	54%	38%	n/a	n/a	n/a	13			Xilinx	XEPLD	3.1				X		
Benchmark #6																				
Xilinx	3090-125	10	0	88%	0%	17	[illegible]	17	15		X	Xilinx	ADI	3.31	X					
Xilinx	3090-125	20	0	100%	0%	9	[illegible]	10	10	X		Xilinx	ADI	3.31	X					
Xilinx	3090-125	10	0	91%	0%	20	21	21	17		X	Xilinx	ADI	3.31				X	X	1
Xilinx	3090-125	20	0	100%	0%	11	11	11	10	X		Xilinx	ADI	3.31				X	X	1
Xilinx	3190-3	10	0	88%	0%	27	31	28	23		X	Xilinx	ADI	3.31	X					
Xilinx	3190-3	20	0	100%	0%	17	18	17	17	X		Xilinx	ADI	3.31	X					
Xilinx	3190-3	10	0	91%	0%	34	36	35	[illegible]		X	Xilinx	ADI	3.31				X	X	1
Xilinx	3190-3	20	0	100%	0%	19	[illegible]	19	17	X		Xilinx	ADI	3.31				X	X	1
Xilinx	4010-5	39	0	87%	0%	25	37	31	24			Xilinx	PPR	1.3	X		X			
Xilinx	4010-5	40	0	90%	0%	32	39	38	[illegible]		X	Xilinx	PPR	1.3			X	X	X	1
Xilinx	4010-5	50	0	100%	0%	29	31	31	21	X		Xilinx	PPR	1.3			X	X	X	1
Xilinx	7272-25	4	0	100%	0%	14	14	14	12			Xilinx	XEPLD	3.1	X					
Xilinx	7272-25	4	0	100%	0%	21	21	21	16			Xilinx	XEPLD	3.1				X		
Benchmark #7																				
Xilinx	3090-125	16	0	100%	0%	20	29	23	[illegible]		X	Xilinx	ADI	3.31	X					
Xilinx	3090-125	20	0	100%	0%	13	15	14	14	X		Xilinx	ADI	3.31	X					
Xilinx	3090-125	12	0	90%	0%	29	32	30	27		X	Xilinx	ADI	3.31				X	X	1
Xilinx	3090-125	[illegible]	0	100%	0%	18	18	18	15	X		Xilinx	ADI	3.31				X	X	1
Xilinx	3190-3	16	0	100%	0%	[illegible]	51	40	33		X	Xilinx	ADI	3.31	X					
Xilinx	3190-3	20	0	100%	0%	24	29	26	25	X		Xilinx	ADI	3.31	X					
Xilinx	3190-3	12	0	90%	0%	47	55	[illegible]	42		X	Xilinx	ADI	3.31				X	X	1
Xilinx	3190-3	20	0	100%	0%	31	31	31	25	X		Xilinx	ADI	3.31				X	X	
Xilinx	4010-5	38	0	86%	0%	19	40	34	38			Xilinx	PPR	1.3	X	X	X			
Xilinx	4010-5	40	0	90%	0%	41	43	42	43			Xilinx	PPR	1.3		X	X	X	X	1
Xilinx	7272-25	4	0	89%	0%	40	40	40	[illegible]			Xilinx	XEPLD	3.1	X		X			
Benchmark #8																				
Xilinx	3090-125	13	0	98%	0%	27	54	41	17			Xilinx	ADI	3.31	X	X				
Xilinx	3090-125	13	0	98%	0%	58	58	58	27			Xilinx	ADI	3.31				X	X	1
Xilinx	3190-3	13	0	98%	0%	47	79	68	38			Xilinx	ADI	3.31	X					
Xilinx	3190-3	13	0	98%	0%	87	[illegible]	[illegible]	38			Xilinx	ADI	3.31				X	X	1
Xilinx	4010-5	38	0	86%	0%	19	40	34	38			Xilinx	PPR	1.3	X	X	X			
Xilinx	4010-5	40	0	90%	0%	[illegible]	66	[illegible]	46			Xilinx	PPR	1.3		X	X	X	X	1
Xilinx	7272-25	4	0	89%	0%	40	40	40	25			Xilinx	XEPLD	3.1	X		X			
Benchmark #9																				
Xilinx	3090-125	16	0	80%	0%	26	38	30	13		X	Xilinx	ADI	3.31	X					
Xilinx	3090-125	20	0	100%	0%	22	38	29	13	X		Xilinx	ADI	3.31	X					
Xilinx	3190-3	16	0	80%	0%	44	71	54	[illegible]		X	Xilinx	ADI	3.31	X					
Xilinx	3190-3	20	0	100%	0%	38	[illegible]	54	21	X		Xilinx	ADI	3.31	X					
Xilinx	4010-5	40	0	100%	0%	41	53	47	[illegible]			Xilinx	PPR	1.3	X	X				
Xilinx	7272-25	8	0	100%	0%	40	40	40	25			Xilinx	XEPLD	3.1	X					

NOTES: 1) Minor routing changes only.

Crit: Criticality used, HM: Hard Macros used, Manp: Manual placement, Manr: Manual routing.

Serial Configuration PROMs

Xilinx offers several pin- and function-compatible serial one-time-programmable PROMs in plastic and ceramic packages.

Presently being shipped are the **XC1736A, XC1765, and XC17128.** (The numbers following the "17" indicate the capacity in kilobits.)

The XC1736A is the only serial PROM that lacks the programmable Reset option; the XC17128 is the only serial PROM that can be clocked at the full 10 MHz required by the XC4000 in fast configuration mode. All other serial PROMs can be clocked at up to 5 MHz.

In early 1993, Xilinx introduced the "D-series" of serial PROMs, the **XC1718D, XC1736D, and XC1765D,** all with programmable Reset polarity, improved ESD protection, and all with max 5 MHz clock frequency. These devices are programmed with a lower voltage and a different programming algorithm than the older parts. The user needs the appropriate update from the programmer vendor. These devices will become the mainstream serial PROMs, and, beyond the traditional packages, they will also be available in the space-saving SO8 package.

In early 1993, Xilinx also introduced the "L-series" of serial PROMs, the **XC1718L and XC1765L**. These devices operate at the new industry-standard low supply voltage of 3.3 V (3.0 to 3.6 V). They can be programmed the same way as the "D-series" parts.

The table at right lists all Xilinx FPGAs and the number of bits required to configure each.

LCA Outputs

XC2000

Complementary outputs, 4 mA guaranteed sink and source current
no slew-rate control.

XC3000

Complementary outputs, 4mA guaranteed sink and source current
slew-rate control adds significant "hesitation" delay.

XC3100

Complementary outputs, 8mA guaranteed sink and source current
slew-rate control adds very little "hesitation" delay.

XC4000

Totem-pole, n-channel-only outputs, 12 mA sink, 4 mA source
slew-rate control similar to XC3100, but reduced "hesitation" delay

XC4000A

(CLB1, reduced routing & cost, increased sink current)
Totem-pole, n-channel-only outputs, 24 mA sink, 4 mA source
Four modes of slew-rate control: FAST, MEDIUM FAST, MEDIUM SLOW, SLOW.
The four modes have the obvious impact on the falling slew rate,
but the rising slew rate is slowed down only in SLOW mode

XC4000H

(Double the number of I/O, without flip-flops)
Totem-pole, n-channel-only outputs, 24 mA sink, 4 mA source, with programmable optional p-channel pull-up transistor. (CMOS output)
Sophisticated, simple slew-rate control, effective only on the falling edge
Resistive: Gradual turn-on, then stay on for continuous 24-mA sink
Capacitive: Gradual turn-on, then gradual turn-off when output is below one threshold voltage. This assures "soft landing", avoids ground bounce. Continuous sink current in capacitive mode is reduced to 4 mA.

General

All LCA outputs have low-impedance protection clamp diodes to GND and Vcc.
Passive resistors:
XC2000: during configuration only (all families use pull-ups during config.)
XC3000: pull-up programmable for inputs only, not for 3-state outputs.
XC4000, all flavors: pull-up or pull-down, progr. for any input or output.

Device	Config. Bits	Device	Config. Bits
XC2064 (L)	12,038	XC4002A	31,668
XC2018 (L)	17,878	XC4003A	45,676
		XC4003H	53,967
XC3020 (A,L)/3120	14,819	XC4004A	62,244
XC3030 (A,L)/3130	22,216	XC4005A	81,372
XC3042 (A,L)/3142	30,824	XC4005/4005H	95,000
XC3064 (A,L)/3164	46,104	XC4006	119,832
XC3090 (A,L)/3190	64,200	XC4008	147,544
XC3195	94,984	XC4010	178,136
		XC4013	247,960

LCA Output Characteristics

Here are the first results of our output characteristics plotting.

Note that one device always represents a whole family, there is no difference between, e.g., XC3142 and XC3190 outputs.

Note that the XC4000 has n-channel-only outputs that do not drive any current above 3.5 V.

When pulling a Low output slightly below Ground, or a High output slightly above Vcc, the output impedance is the same as it is on the other side of Ground and Vcc, i.e., the plot shows a straight line going through Ground and Vcc. (The current direction changes, of course.)

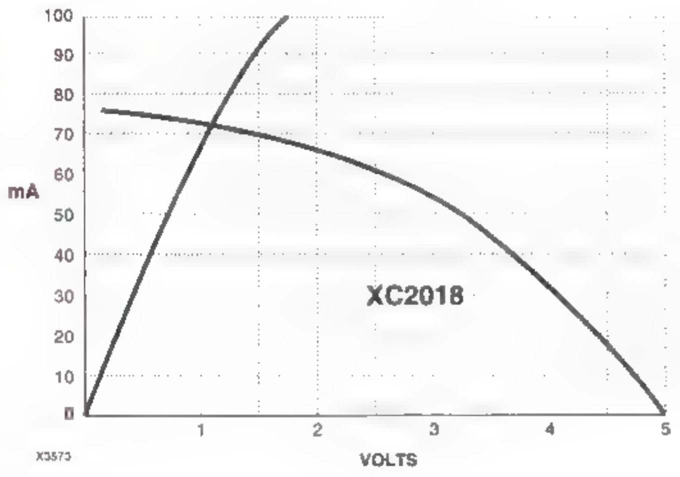
When the voltage exceeds 0.5 V below Ground or 0.5 V above Vcc, the protective diodes become conductive, and the current increases **dramatically**. That's why we should not specify a max voltage excursion, but rather a max current excursion into the forbidden territory below ground and above Vcc.

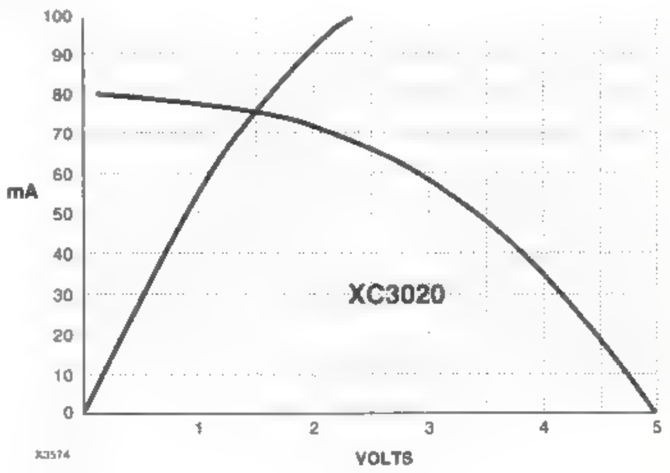
This is true for all devices. **Even XC4000 outputs have a strong clamp diode against Vcc.** Disregard previous statements to the contrary.

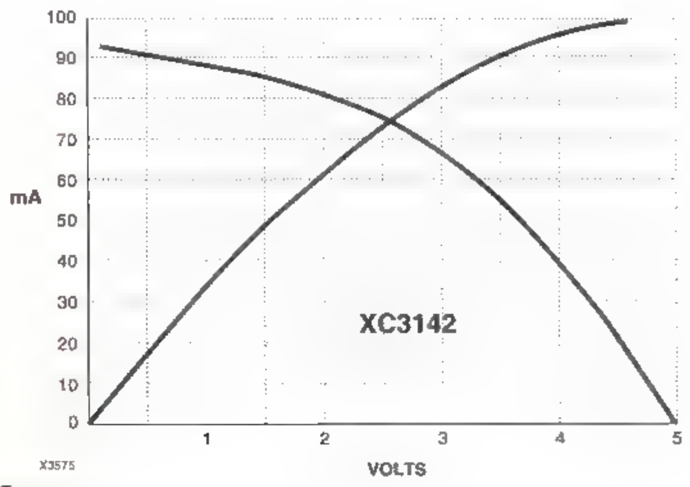
All measurements at 25°C and Vcc=5.00 V

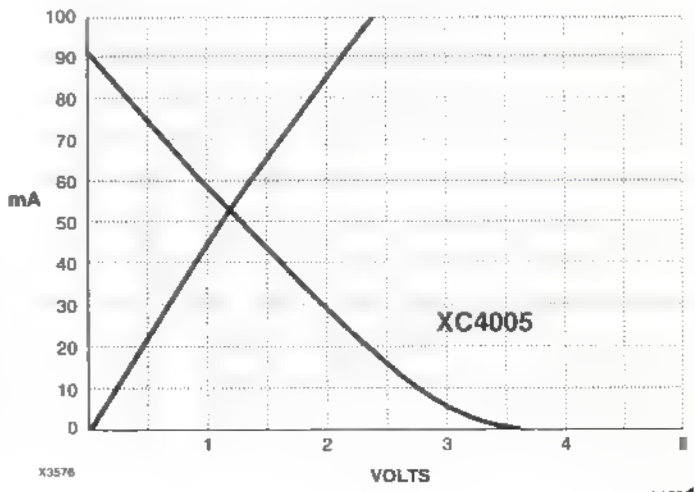
All parts are 1993 production type. PA

Sink Current and Output Low Impedance				Source Current and Output High Impedance			
Device	1 V	2 V	Impedance	4V	3V	2V	Impedance
XC2018	70	>100 mA	14 Ω	-30	-52	-65 mA	35 Ω
XC3020	55	100 mA	20 Ω	-35	-60	-75 mA	30 Ω
XC3142	35	63 mA	30 Ω	-35	-60	-73 mA	30 Ω
XC4005	42	85 mA	23 Ω	0	-7	-30 mA	40 Ω









XC4000H Output Rise and Fall Times

Slew-Rate Control

The XC4000H outputs use a novel, patent-pending method of slew-rate control that reduces ground bounce with no significant delay penalty. Each output has two slew-rate options, both of which reduce the positive ground bounce that occurs when the pull-down transistor is turned on. They differ in the way the output current is turned off.

The slew-rate-limited default mode is called "capacitive," or "SoftEdge." At the beginning of a High-to-Low transition, the pull-down transistor gradually turns on, and is fully conductive until the output voltage has reaches +1 V. The pull-down transistor then partially turns off, so that its final on-resistance of about 100 Ω, low enough to sink 4 mA continuously. Gradually turning off the transistor reduces the rate of current decrease (di/dt) that is responsible for negative ground bounce.

The "capacitive," or "SoftEdge," mode is the best choice for outputs requiring less than 4 mA of the dc sink current.

The non-slew-rate limited mode is called "resistive," and differs from the "capacitive " mode in that the pull-down transistor remains fully conductive as long as the output is Low. The pull-down transistor has an on-resistance of <20 Ω, and is capable of sinking 24 mA continuously.

Resistive mode is required for driving terminated transmission lines with 4 to 24 mA of dc sink current. The abrupt current change when the output voltage reaches zero causes a negative voltage spike over the ground inductance (bonding wire). This ground bounce can become objectionable when many outputs switch High-to-Low simultaneously.

The figures below show rising and falling output edges for various loads. The upper trace in each figure shows a second output driven from the same internal signal, but unloaded. It acts as a timing reference, and triggers the oscilloscope.

Resistive mode and capacitive mode transitions start with similar internal logic delays. Resistive mode falls faster, and has more undershoot; capacitive mode rises slightly faster. For a 200-Ω pull-up, 330-Ω pull-down termination, only resistive mode is meaningful. A TTL output with a 1000-Ω pull-up, 150-pF termination has a slow (150 ns) final rise time that extends beyond the 10-ns shown.

Summary

Use resistive mode for applications that require >4 mA of dc sink current, and for heavy capacitive loads when they must be discharged fast. Use capacitive mode for all other applications, especially light capacitive loads (50 to 200 pF) and all timing-uncritical outputs that require < 4 mA dc current. The Low-to-High transition is not affected by the choice of slew-rate mode.

The figures below are the unedited results of tests performed on a multi-ground-plane test PC board, manufactured by Urban Instruments (Encino, CA). Measurements were made with a Tektronix TDS540 digital storage oscilloscope. The time scale is 2 ns/division.

Trace A shows Resistive mode with CMOS outputs

Trace B shows Resistive mode with TTL outputs

Trace C shows Capacitive mode with CMOS outputs

Trace D shows Capacitive mode with TTL outputs

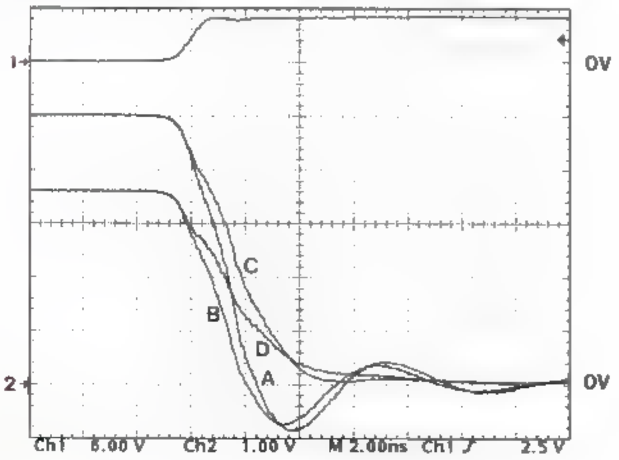


Figure 1. Falling Edge, 50 pF Load

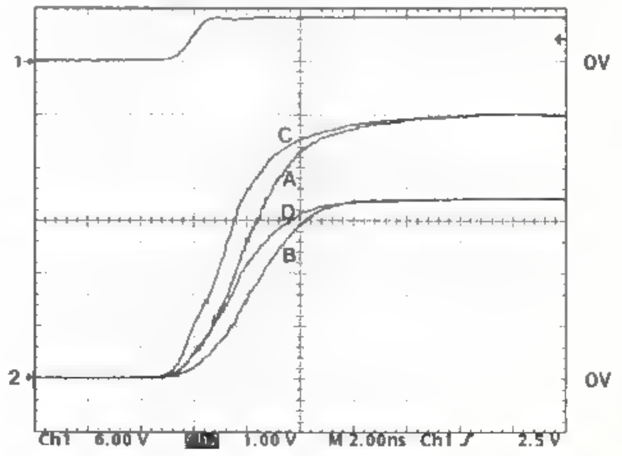


Figure 2. Rising Edge, 50 pF Load

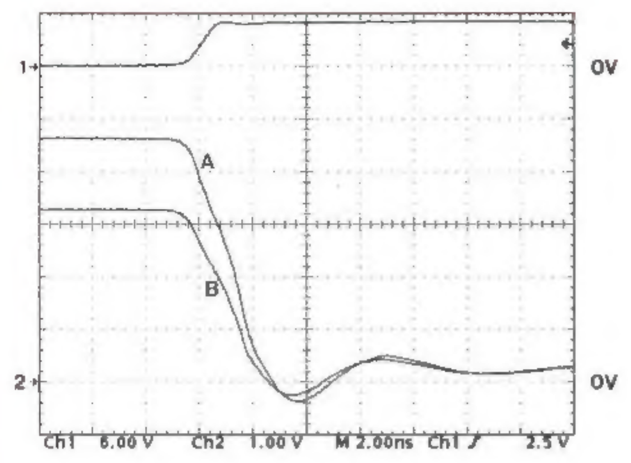


Figure 3. Falling Edge, 200/330 Ω, 50 pF Load

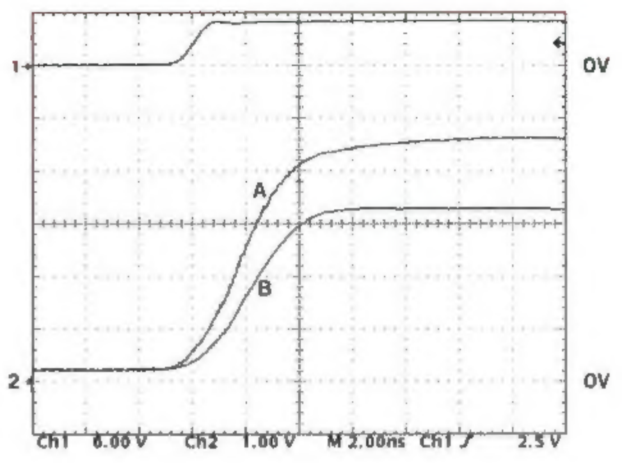


Figure 4. Rising Edge, 200/330 Ω, 50 pF Load

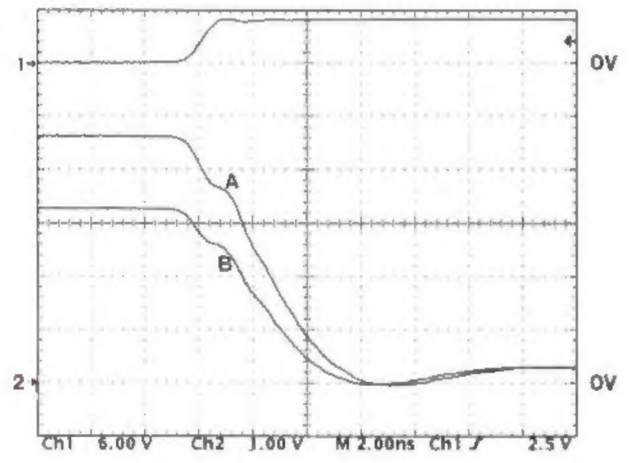


Figure 5. Falling Edge, 200/330 Ω, 150 pF Load

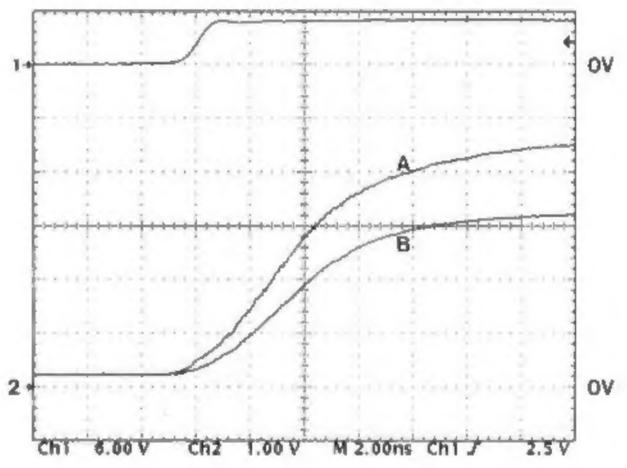


Figure 6. Rising Edge, 200/330 Ω, 150 pF Load

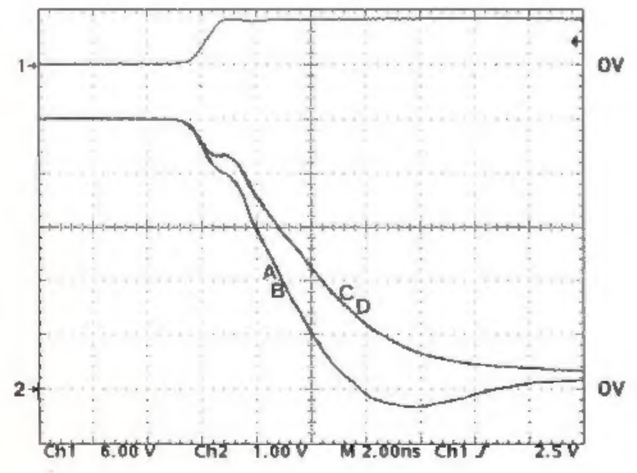


Figure 7. Falling Edge, 1000 Ω, 150 pF Load

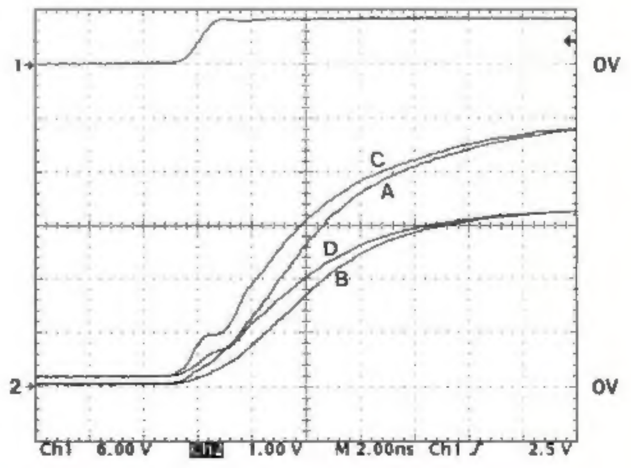


Figure 8. Rising Edge, 1000 Ω, 150 pF Load

Two's Complementer Packs 2 Bits per CLB

The best known algorithm for twos complementing a number is to invert all the bits and then add one to the result. Using this algorithm, only one data bit can be generated by each XC3000 CLB, since the increment operation requires an additional carry output for each bit. However, an alternate empirical algorithm exists that does not have this limitation, and generates two bits per CLB.

This alternate algorithm permits the two's complement of a number to be determined by inspection. As shown in the example, the number is scanned, one bit at a time, from the least significant end, until the first "one" is encountered. The first "one" and any less significant zeros are passed to the output unchanged. All more significant bits are inverted.

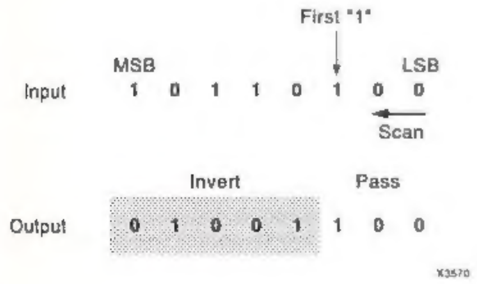
This algorithm may be rewritten in an iterative form: a bit is inverted only if its less significant neighbor is inverted, or is a one. Trailing zeros and the first "one" are not inverted because their less significant neighbors are neither inverted nor "ones". The bit after the first "one" is inverted because its neighbor is a "one", and the remaining bits are inverted because their neighbors were .

This may be implemented as shown in Figure 1. The inputs and outputs of the less significant neighbors are inspected to determine their value and whether they were inverted. An XOR is then used to invert the data according to the rule described above.

The least significant bit always remains unchanged when two's complementing a number. Consequently, no logic is required by the LSB and no less significant neighbor is required.

Figure 2 shows a modification of the 2-bit CLB that only incurs one delay per bit-pair. This doubles the performance of the original design, without increasing the number of CLBs required or the routing complexity.

A further modification, not shown, permits the delay for an 8-bit complementer to be reduced to two CLBs. However, one additional CLB is required. In this design, complementers larger than eight bits use two additional CLBs per three bits, and the delay increases by one CLB per three bits.



Two's Complement Example

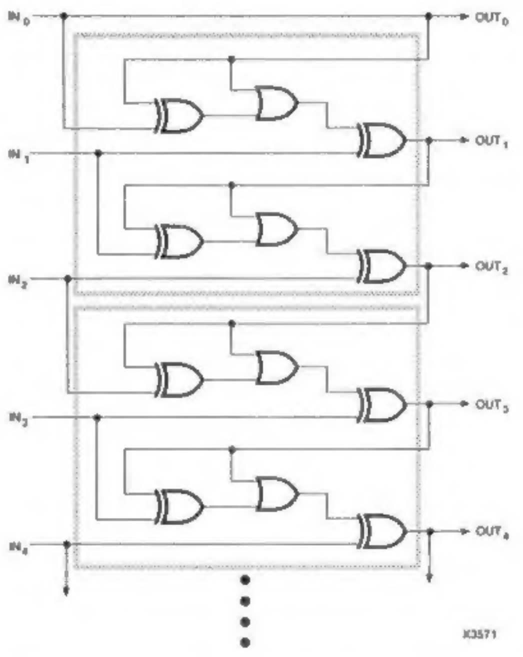


Figure 1. Simple Two's Complementer

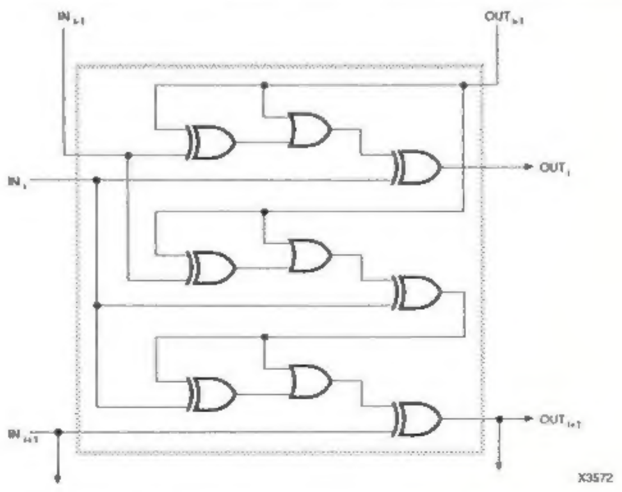


Figure 2. Faster Two's Complementer

Excessive Idle Power in XC2000

Some users report a quiescent Icc consumption of more than 10 mA in the XC2000 family. This is usually the result of floating input pads, especially unbonded ones, and it can be fixed quite easily.

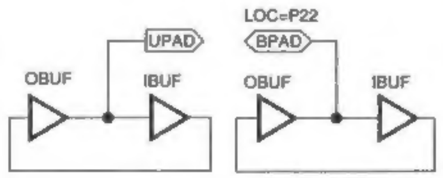
While the XC3000 and XC4000 devices have default pull-up resistors on all inputs, the XC2000 family lacks this option. Each **unused** pin or pad must, therefore, be forced to a valid logic level, either by an external connection or resistor, or by using its own output driver. The Makebits Tie option does **not** take care of this, it only ties **internal** inputs and interconnects to a defined logic level.

Users tend to overlook the **unbonded** pads on XC2064 PC44, XC2018 PC44 and XC2018 PC68. These devices have more internal pads than there are package pins available. Some pads are, therefore, left unbonded, but these unbonded inputs must also be forced to a valid logic level. Otherwise they might cause uncontrolled power consumption, and even uncontrolled oscillations. Unused outputs, bonded or not, can be tied from the schematic diagram. To do this, create dummy bidirectional pins using OBUFs, IBUFs, and BPADs or UPADs, as appropriate. Connect the OBUF output directly to IBUF input. This connection creates a tie circuit that is an input-less latch with no input. After configuration, these tie-circuits attain an unspecified, but well-defined logic level, and remain there, thus preventing the input from floating.

In existing designs, tie-circuits locked to unused pins can be added to the schematic, which is then recompiled using the previous LCA file as a guide. In new designs, an appropriate number of bonded-pin tie circuits included in the schematic will automatically be distributed among the unused bonded pins.

Tie-circuits cannot be locked to specific unbonded pins. However, if the correct number of unbonded tie circuits are included in the schematic, all unbonded pads will be tied. Tie circuits may also be added by editing the LCA design in XDE.

Unused bonded outputs tied in this way are active pins, and cannot, therefore, be used as PCB feedthroughs. However, unused pins required as feedthroughs need not be tied in the LCA device, since they do not float. PA



X3607

Don't Drive Mode Pins from Uncontrolled Sources

We recently debugged a design with unpredictable power-up behavior. The designer had used an Altera EPLD device to control the M0, M1, M2, and $\overline{\text{PWRDWN}}$ pins of an XC3042. (M0 and M1 were used for readback. We don't know why an EPLD powerhog was used to control the LCA power.)

Upon power-up, the EPLD puts uncontrolled signals on its outputs, lasting for up to 100 ms. That's long enough to make the XC3042 configure in the wrong mode, become a master instead of a slave, thus crashing the system. Remember, a Low on $\overline{\text{PWRDWN}}$ causes all inputs, including $\overline{\text{RESET}}$ to be interpreted as High.

Connecting external logic to the mode and $\overline{\text{PWRDWN}}$ inputs of an LCA must be done with care and a thorough understanding of power-up conditions. Xilinx FPGAs have been carefully designed to avoid erroneous output signals during power-up. Other ICs are not necessarily that friendly. PA

No Power Penalty for Slew-Rate Limiting

When outputs are configured as slew-rate limited, they charge and discharge their capacitive loads more slowly, thus generating less ground bounce. The question has been raised whether that increases power consumption. It does not.

When an empty capacitor is charged through any resistor, large or small, linear or non-linear, half the consumed anergy ($1/2\ CV^2$) ends up in the capacitor, while the other half heats up the resistor, whatever its value may be.

Cycling a capacitor at a given frequency, therefore, consumes the same power, $P = f\ C\ V^2$, irrespective of the series resistor, provided it is small enough to allow a fairly complete charge and discharge cycle.

Simple law of physics. PA

Xilinx Training Classes

Xilinx Training Classes allow you to get up-to-speed on the latest Xilinx products quickly and efficiently. If you are new to Xilinx, or would like to learn more about the capabilities of the products, the classes are for you. Hands-on interactive instruction is the best method of learning how to use the software effectively. Benefits include:

- Start or improve your design during the class
- Reduce your learning time
- Make fewer design iterations
- Get to market faster
- Lower production costs
- Increase quality

Xilinx offers a variety of classes worldwide at the locations listed below. We can also bring classes to your own facility. Call your closest Xilinx sales office or Regional Training Center for more information.

United States

San Jose, CA	(408) 879-5090
Sacramento, CA	(916) 781-6614
Woodland Hills, CA	(714) 641-4198
Costa Mesa, CA	(714) 641-4198
San Diego, CA	(619) 571-7540
Portland, Oregon	(503) 526-6217
Seattle, Washington	(206) 881-6697
Salt Lake City, Utah	(408) 743-3463
Denver, Colorado	(303) 799-7820
Phoenix, Arizona	(602) 961-6403
Arlington, Texas	(214) 960-1043
Minneapolis, MN	(612) 932-0678
St. Louis, Missouri	(708) 860-8559
Chicago, Illinois	(708) 860-8517
Detroit, Michigan	(708) 860-8559
Cleveland, Ohio	(216) 573-7400
Boca Raton, Florida	(408) 879-5090
Baltimore, MD	(410) 684-6683
Westboro, MI	(508) 870-0312
Rochester, NY	(408) 879-5090

International

Hatfield, UK	(44) 707 284 154
Paris, France	(33) 1 45 92 65 00
Lisbon, Portugal	(35) 1 3100 000
Munich, Germany	(49) 8142 51019
Freiburg, Germany	(49) 761 405111
Breda, Netherlands	(31) 76 784911
Switzerland	(41) 61 975 00 00
Pretoria, Republic of South Africa	(27) 1273-1122
Bombay, India	(91) 212 332 838
Beijing, People's Republic of China	(86) 1-256 1155 x2357
Hong Kong	(852) 410 2780
Tokyo, Japan	(81) 3-3566-6364

XILINX®

2100 Logic Drive
San Jose, CA 95124-3450

U.S. POSTAGE
PAID
FIRST CLASS
PERMIT NO. 2196
SAN JOSE, CA